Scalable emulation of dynamic multi-hop topologies

Nickelsen, Anders; Jensen, Morten N.; Matthiesen, Erling Vestergaard; Schwefel, Hans-Peter

Published in:
The Fourth International Conference on Wireless and Mobile Communications, 2008. ICWMC '08

DOI (link to publication from Publisher):
10.1109/ICWMC.2008.44

Publication date:
2008

Document Version
Early version, also known as pre-print

Scalable emulation of dynamic multi-hop topologies

Anders Nickelsen †, Morten N. Jensen †, Erling V. Matthiesen †, Hans-Peter Schwefel †‡
† Networking and Security Section, Department of Electronic Systems, Aalborg University, Aalborg,
‡ Forschungszentrum Telekommunikation Wien - FTW, Vienna, Austria
E-mail: {an,mnje03,evm,hps}@es.aau.dk

Abstract

Communication in wireless networks is affected by uncontrollable disturbances in the channel. Effects of these disturbances are exacerbated in networks with dynamic topologies and multiple hops. Lack of control of the channel complicates testing of such networks as test conditions are hard, or impossible, to reproduce. This paper describes how to create reproducible test conditions for these networks by emulating the wireless links. Emulation is performed by a topology emulator to which end-nodes are connected using wired links. In real-time, the emulator imposes packet drops and delays onto traffic between end-nodes. The imposed properties are based on simulations of node mobility, loss and delay models. Evaluation confirms that the testbed is capable of emulating links in real time and transparent to upper layer protocols. Additionally, the impact on test results is evaluated, such as increased network delays and reduction of bandwidth when loading the emulator. Finally, an outlook on advancing capabilities and how to integrate such in the emulator is presented.

1. Introduction

Wireless technologies are deployed in increasingly many types of mobile devices. The vast popularity of these mobile devices, and thus the increased availability of wireless technologies, makes applications for the mobile domain of great interest. Evaluation is an integral part of developing such applications. Several evaluation methods apply, such as simulation, emulation or experimental tests. Simulation tools, such as NS-2 [7], provide a high level of detail in the networking layers. The application under test is, however, typically simplified as the real application cannot easily be used in the simulated environment. Hence, the simulation results represent an application model and not of the application itself. Experimental setups can be used to test the real application. However, disturbances in the environment may have a huge impact on the test results. Moreover, the characteristics of these disturbance are not always controllable, making it very difficult to repeat even simple test runs [10]. Emulation provides a hybrid of simulation and experimental setups by allowing real applications to be subjected to simulated network conditions. The control of the simulation allows for accurate reproductions of the test conditions to repeat test runs. A disadvantage of using emulation is that real applications operate in real time and thus enforce the network emulation to also effectuate in real time.

The objective of this work is to develop an emulation testbed for wireless applications, where the wireless link is replaced by a wired link. The wired link is less exposed to uncontrollable disturbances and thus more controllable than a wireless link. The properties of a wireless link, such as *packet drops* and *delay*, are then enforced on network traffic on the wired link in a controllable and reproducible manner based on simulations.

Several requirements must be fulfilled in order to facilitate emulation. Transparency of the emulation to the network application is inherent to create results comparable to those of experimental setups. Transparency regards performance, which must resemble a real wireless setup and network protocol interfaces. These interfaces must present features to the upper layers similar to a the real setup. Moreover, accurate communication models must be used in simulation to resemble influences from a real environment. Lastly, simple interfacing in terms of deployment, use and result processing is required to ensure usability of the emulator.

The contributions of this paper are: 1) description of the design of the topology emulator, a testbed capable emulating dynamic multi-hop topologies by changing time-varying link properties in real-time and 2) evaluations of the topology emulator to illustrate that is capable of emulating dynamic multi-hop topologies while accomplishing the specified requirements.

1.1. Related work

Many solutions handle network emulation by manipulating the same properties as in the proposed solution. An extension to ns-2, called ns-2e [6], is capable of emulating network properties from a simulated ns-2 network model by centrally affecting network traffic. The approach of Seawind [5] is similar. The challenge in these approaches is that the deployment effort increases when scaling the number of end-nodes, as the setup changes when more nodes are needed, for instance by installing several network interfaces. Also the performance of the real-time emulation may degrade resulting in a drifting emulated scenario which does not meet real-time requirements. Opposed to the centralized approach is the distributed approach where each node calculates its own networking properties and this data must be synchronized between all participating nodes. A such approach is used by both EMWIN [11] and JEmu [4]. Testing real applications on such tools becomes a challenge as the end-node environment is deployed as a virtual machine and thus not a native environment. This work provides a scalable centralized environment, for real applications on real end-nodes, while obeying the real-time constraints of online emulation.

2. Topology emulator overview

The topology emulator is designed for scalability using modular network and software architectures, which are described in the following.

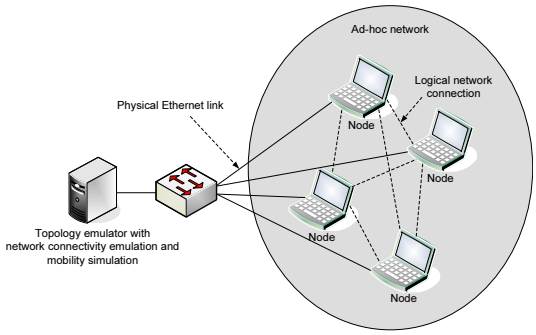

Figure 1. Topology emulator network architecture; emulator node, switch and end-nodes.

2.1. Topology emulator architecture

The network architecture of the topology emulator is illustrated in Figure 1. It consists of one centralized node called the *emulator node* and a central network switch to which all *end-nodes* are connected. End-nodes hold the applications or communication protocol (Layer 3 and higher) to be evaluated. Changing from a wireless to a wired interface on the end-node is the only change from a real setup to using the emulator. The switch facilitates connecting many nodes to the emulator providing a scalable network architecture.

The emulator node receives all frames transmitted between end-nodes. No end-nodes receive frames before they have been bridged by the emulator node. Node separation is obtained through 802.11q virtual LAN (VLAN) tagging [8] on the switch. This concentrates all frames from one end-node into one VLAN unique for that end-node. The VLAN-tag is also used by the emulator node to identify the sources of the frames. By enabling the emulator node to bridge between all virtual LANs, all end-nodes can successfully transmit frames to each other, given they are forwarded by the emulator node. By selectively dropping or delaying frames, the emulator node controls the link properties of all links between end-nodes. Ultimately, as the emulator node forwards frames, its existence is by design transparent to any networking protocols (Layer 3 and higher) used on the end-nodes.

The software architecture of the emulator node consists of two parts, namely a simulation part and an emulation part. The simulation part, detailed in Figure 2, calculates the link properties by simulating node movement and wireless links and saves them into trace-files. The emulation

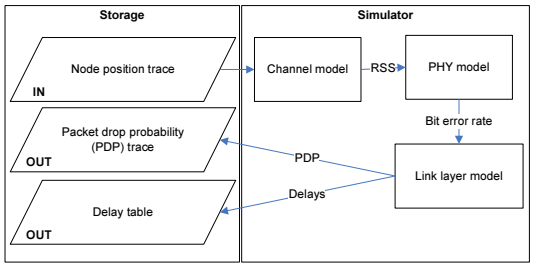

Figure 2. Simulation part of the emulator node. Traces of node movement are transformed into traces of packet drop probabilities and packet delays for the emulation.

part, depicted in Figure 3, emulates the properties of the simulated wireless links real-time while end-nodes communicate. The properties are loaded from the trace-files and used to decide if a packet should be dropped or not. If packets are not dropped the emulator determines a delay for each packet and transmits it once the delay expires.

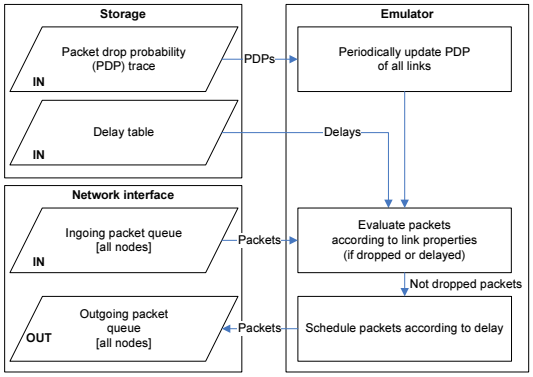

Figure 3. Emulation part of the emulator node. Link properties are used to evaluate incoming packets. Packets that are not dropped are scheduled for delayed forwarding.

2.2. Simulation models

Node mobility is input to the emulator node in the form of node positions over time t as (x, y)-coordinates. These positions are based on realizations of either deterministic paths that nodes follow or based on stochastic models such as random walk or random way-point [2].

Packet drop probability (PDP) on a link is calculated based on distances between nodes and models of the wireless environment. As shown in Figure 2, models of the channel, the physical layer and the medium access control can be used to calculate PDP. The output of the calculations is a trace of probabilities over time $p_{i,j}(t)$ on every link (between nodes i and j) in the topology. Note that the symmetry of the probabilities on a link depends on the loss model, as the topology emulator supports emulating asymmetrical links. Based on a probability threshold P the number of next-hop neighbors $n_i(t)$ of node i at time t is calculated from $p_{i,j}(t)$ for all t. This is used to determine how many nodes share the channel when determining delay for a packet sent by node i at time t.

Delay of a packet at a given time t is determined from two parameters; $n_i(t)$ and a static delay table. Each row in the table contains delays distributed according to the delay model when $n_i(t) + 1$ nodes share the channel. The delay table is calculated in the simulation part prior to emulation. During emulation, the specific delay is determined by drawing a number randomly from the set in the $n_i(t)$'th row. The use of delay tables instead of an analytic model of the delay allows for complex delay assumptions during delay simulation, as a parametric delay distribution does not need to be known to determine a specific packet delay during emulation.

All simulation models used to generate input to the emulator must compensate for the fact that all calculations are performed before running the emulating. In effect, this means for the delay that the amount of and characteristics of traffic used in the emulator must be reflected in the models. Also, it is not possible to dynamically change communication parameters, such as modulation scheme, during emulation. These restrictions must be implemented in the simulation models using constant parameters such as expected network utilization.

2.3. Deployment

All nodes in the topology emulator are connected to a Cisco Catalyst 2950T VLAN-aware switch, equipped with two Gigabit ports. One of these ports is used in trunking mode, meaning that all traffic sent on other ports is forwarded to this port. The emulator node is then connected to one Gigabit port for forwarding packets using *iptables*. The machine is has an Intel Core 2 Duo 1.86GHz processor and has Linux kernel v2.6.18 installed. Once a packet is received on the emulator node, a netfilter kernel module ensures that the link to the receiving node is available ($p_{i,j}(t) < P$). This is to prevent successful transmission of link layer datagrams, e.g from using ARP [9], when there is no emulated link present. Link layer datagrams are not handled in the IP-layer, and thus not by iptables, in Linux. If a link exists, the packet is enqueued by iptables for the emulating process. This process determines if the packet is to be dropped from $p_{i,j}(t)$, which is updated every 100ms. The process also determines the delay by using the number of neighbors, $n_i(t)$, on node i and the delay table. The delay table is a $N - 1 \times 1000$ matrix, where N is the number of nodes connected to the switch. Once the delay has been determined, the packet is scheduled for delayed transmission. The packet scheduler checks for packets to send every $122\mu s$.

3. Evaluation

The evaluation of the topology emulator is two-fold; 1) evaluating that the emulator is capable of emulating dynamic multi-hop topologies and 2) verifying the specified performance requirements. Both evaluations are described below.

3.1. Functionality evaluation

A scenario with mobile nodes, creating a dynamic topology, and communication using ad-hoc routing is used to evaluate the emulator functionality. The node movement is illustrated in Figure 4. During the scenario, a node in

the end-to-end path disappears and is replaced by another node, effectively breaking links around the disappearing node temporarily. Throughout the scenario the link is subjected to packet drops and delays from the simulations. The used models are of an IEEE 802.11g wireless link incorporating shadowing and fading in the channel. The objective of the scenario is to illustrate how the ad-hoc routing algorithm is subjected to a dynamic multi-hop topology. The

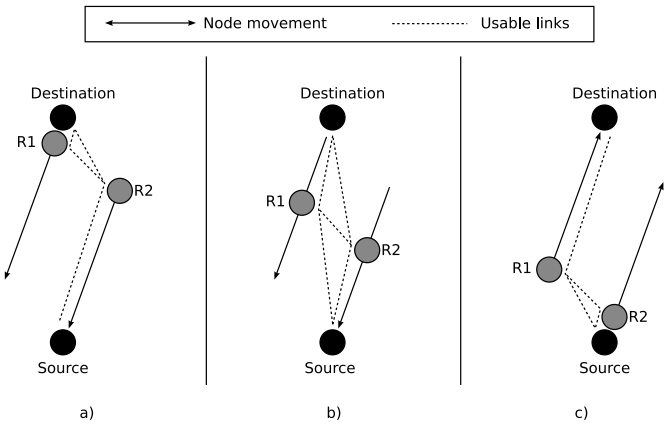

Figure 4. Movement scenario with relay nodes R1 and R2 move periodically back and forth between Source and Destination, allowing changing 2-hops end-to-end paths and persistent 3-hops paths.

specified movement is used as input to simulate PDP and delay. The PDP of the links is illustrated in Figure 5 as the probability varying over time on each link in the network topology. A 3-hop path is always present (R2-R1-Destination), as is a 2-hop path. However, the specific links of a 2-hop path change over time. This effectively illustrates the dynamic multi-hop topology. The trace of simulated PDP and the delay table are then used as input for the emulation process to impose the topology information onto real traffic. *ping* is set to continuously ping from source to destination while OLSR [3] is used as ad-hoc routing algorithm between the nodes. The resulting traffic recorded by the emulator is illustrated in Figure 6.

From the figure, we see that the emulator is capable of subjecting end-nodes to a dynamically changing multi-hop topology. This is seen as the flow of packets is redirected to use the available links when the currently used link becomes unavailable. Moreover, the packets sent using the R1-R2 link clearly illustrates the multi-hop emulation capability.

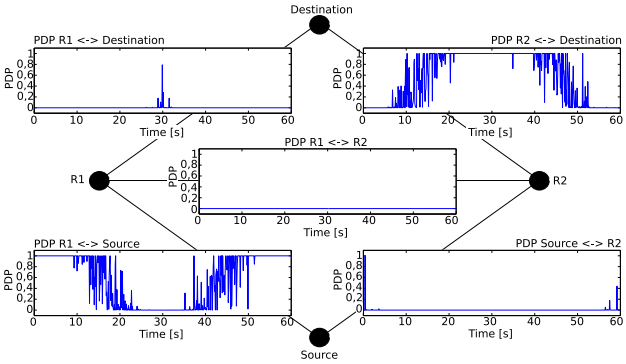

Figure 5. PDP trace on links in the evaluation scenario. The end-to-end path changes from using *R2-Destination*-link to using *Source-R1*-link and back over 60 seconds.

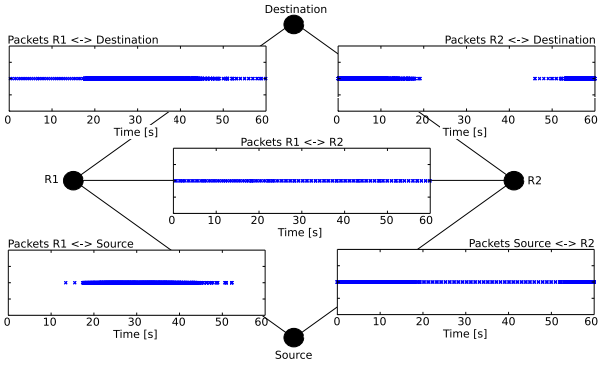

Figure 6. Traffic on links during emulation as recorded by the topology emulator shows that only available links are used, which change over time.

3.2. Performance verification

To ensure that the emulator performs as required, evaluations of bandwidth and service time have been performed. As previously described, the emulator must appear transparent to the end-nodes meaning that decreased network bandwidth and increased link delay (besides the emulated delay) is not tolerated when comparing to a wireless environment.

Bandwidth limitations in a network occur at processing bottlenecks. In the emulator, two such potential bottlenecks exist, i.e. in the switch and in the emulator node. As the complexity and processing need in the emulator node is far greater than that of the switch, the emulator node is considered the significant bandwidth bottleneck in the setup. Hence, the bandwidth capabilities of the emulator node are

evaluated. To evaluate available bandwidth of the emulator node, a traffic generator called D-ITG [1] is used. By use of 4 nodes in a fully connected emulated topology, each sending and receiving a stream of totally 100MBit/s asynchronously distributed to the 3 other nodes, the emulator node is loaded heavily. D-ITG is then capable of recording the received bandwidth by the nodes, which during successful emulation should amount to 100Mbit/s per node, deducting a small overhead percentage from transport and network layers. In total this amounts to 400Mbit/s load on the emulator.

The results of the bandwidth test show that all nodes receive the expected bandwidth of 100Mbit/s, meaning that emulator is capable of handling a bandwidth of at least 400Mbit/s. Considering that the expected throughput attained with IEEE 802.11g in a real wireless channel is 54Mbit/s, the emulator node is capable of supporting 8 separate channels within the 400Mbit/s limit. Note however, that this limit is not of the topology emulator, but of the testing equipment and that the actual bandwidth limit of the emulator node itself may be higher. Note also, that the derived limitation of 8 channels is equal to 16 connected, fully loading nodes in separated groups of two and is in effect only when the nodes experience a traffic-free channel. This is of course a possible situation when testing the wireless application, however, not considered to be very likely. Therefore, the bandwidth capabilities of the emulator node are considered acceptable.

Service time (emulator processing delay) is also evaluated by use of *ping* by measuring the round-trip time for a link by sending out packet probes. The evaluation is performed on a emulated link (with emulated delay = 0) and compared to both a direct wired link and measurements of an IEEE 802.11g wireless link. This is done to establish the service time of the emulator node to learn if it is capable of emulating even the smallest delays expected in a real wireless connection. As the service time relates to processing time of a packet, and this time is dependent on the packet size, several packet sizes are used. Packet of sizes 0-5000 bytes, with standard Ethernet frame size of 1500 bytes payload are used to show if the service time of the emulator compare to a direct wired and a direct wireless link. The results of the service time measurements are illustrated in Figure 7, where we see that the emulator node uses approx. $250\mu s$ more to service a packet than when using a real wired link. In addition, the figure shows that these excess $250\mu s$ are well below the measurements of a wireless link. This means that the emulator processing delay is acceptable as it is below the values of an IEEE 802.11g link.

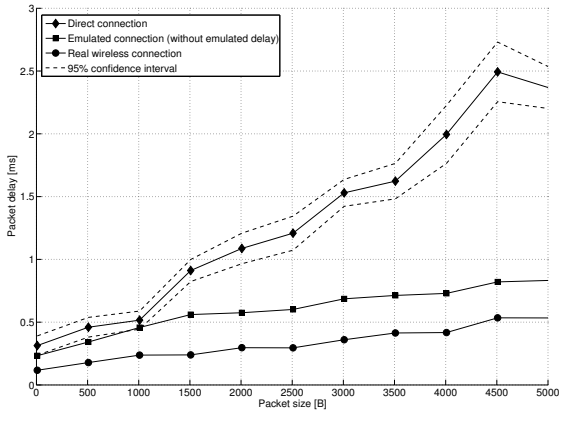

Figure 7. Service time from various packet sizes on: A real wireless IEEE 802.11g link, a direct emulated link (with delay = 0) and a direct wired Ethernet link. The confidence interval is shown for the wireless measurements only, as the variations in the remaining measurements are insignificant.

4. Limitations

The software architecture of the topology emulator is developed using a two-stage model divided between a simulation part and an emulation part. On one hand, this model enables complex link simulations to be performed free of real-time constraints and the emulation to be performed with minimized computation during real-time constrained run-time. On the other hand, the two-stage model prohibits any changes in movement or traffic to affect the link models during run-time, as the link properties are only calculated once prior to emulation. Also, as all emulation is carried out on the emulator node, end-nodes have no way of determining position or measuring link quality. Finally, only PDP and delay have been calculated, meaning that the upper layers do not have access to traditional link layer information such as received signal strength indication (RSSI). The setup requires all upper layer technologies to be independent of the link layer and the physical layer. Affecting the simulation models from the emulation can be approached by simulating several sets of link properties, based on real-time parameters such as network utilization, and then choosing sets dynamically during emulation. Furthermore, the simulation can be performed during emulation, and thus use real parameters, for instance to allow node movement to depend on link quality. This approach is currently being investigated for the emulator.

5. Conclusions and future work

In this paper, we present a new network emulating tool capable of emulating dynamic multi-hop topologies. This is especially important when developing wireless applications, as field tests of such applications become cumbersome or even impossible to reproduce. As an advance to existing tools, the topology emulator features real-time emulation of dynamically changing multi-hop topologies that are resulting from node movement in a pre-specified scenario. Moreover, the architecture of the topology emulator is designed to be scalable and modular to facilitate extensions without any modification to the deployed version.

The emulation functionality is two-stage; a simulation part and an emulation part. The simulation part simulates a complex wireless network from node movement resulting in packet drop probabilities and delays on each link between nodes. The emulation part executes these properties real-time on a central emulator node. End-nodes, that are usually in the wireless domain, are connected to the emulator node via wired links through a central switch. All frames sent from end-nodes are forwarded to the emulator node and based on the link property traces the emulation part decides if frames should be forwarded further. If so, a delay is determined and the frames are scheduled for transmission to the receiving end-node.

By designing the connecting point as a switch, the network architecture allows for up to 20 real end-nodes to be connected to the emulator. Evaluations of the functionality and the performance of the emulator shows that it is capable of emulating dynamically changing topology properties transparently towards all connected end nodes. The results also show, that the nodes do not experience bandwidth limitations from using the emulator. Moreover, the delay experienced from the processing of the emulator is acceptable as it is well below the actual delays to be emulated, and are as such handled inside the topology emulator.

Limitations of the two-stage approach have been addressed and work is ongoing exploring the options for enabling the nodes to affect the simulation properties. Also, future work includes dedicated performance evaluations to research where the actual performance limit of the topology emulator lies.

6. Acknowledgments

This work was partially supported by the EU IST FP6 project 'HIghly DEpendable ip-based NETworks and Services – HIDENETS', see www.hidenets.aau.dk. The Telecommunications Research Center Vienna (ftw.) is supported by the Austrian Government and by the City of Vienna within the competence center program COMET.

References

[1] S. Avallone, A. Pescape, and G. Ventre. Distributed internet traffic generator (D-ITG): analysis and experimentation over heterogeneous networks. *ICNP 2003 poster Proceedings, International Conference on Network Protocols, Atlanta, Georgia*, 2003.

[2] C. Bettstetter, H. Hartenstein, and X. Pérez-Costa. Stochastic Properties of the Random Waypoint Mobility Model. *Wireless Networks*, 10(5):555–567, 2004.

[3] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), Oct. 2003.

[4] J. Flynn, H. Tewari, and D. O'Mahony. A Real-Time Emulation System for Ad Hoc Networks. *Proceedings of the Communication Networks and Distributed Systems Modeling and Simulation Conference*, 2002.

[5] M. Kojo, A. Gurtov, J. Manner, P. Sarolahti, T. Alanko, and K. Raatikainen. Seawind: a Wireless Network Emulator. In proceeding of 11th GI. *ITG Conference on Meaurement, Modelling and Analysis (MMB 2001)*, pages 151–166, 2001.

[6] D. Mahrenholz and S. Ivanov. Real-Time Network Emulation with ns-2. *Proceedings of the The 8-thIEEE International Symposium on Distributed Simulation and Real Time Applications (DS-RT 2004)*.

[7] S. McCanne, S. Floyd, et al. Network simulator ns-2. *The Vint project, available for download at http://www. isi. edu/nsnam/ns*.

[8] D. McPherson and B. Dykes. VLAN Aggregation for Efficient IP Address Allocation. RFC 3069 (Informational), Feb. 2001.

[9] D. Plummer. Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware. RFC 826 (Standard), Nov. 1982.

[10] H. Waeselynck, Z. Micskei, M. Nguyen, N. Riviere, and F. LAAS-CNRS. Mobile Systems from a Validation Perspective: a Case Study. *Parallel and Distributed Computing, 2007. ISPDC'07. Sixth International Symposium on*, pages 14–14, 2007.

[11] P. Zheng and L. Ni. EMWIN:: emulating a mobile wireless network using a wired network. *Proceedings of the 5th ACM international workshop on Wireless mobile multimedia*, pages 64–71, 2002.